RELATÓRIO DO PROJETO DE BIOINFORMÁTICA

SUPERCOMPUTAÇÃO - 24.2

Alexandre Santarossa – 06/12

Introdução

A bioinformática é uma área interdisciplinar que combina biologia, computação e matemática para analisar grandes volumes de dados biológicos, partindo para o contexto da biologia molecular, é fundamental a análise de sequências genéticas, que são compostas por DNA e RNA. Essas sequências possuem as informações necessárias para a síntese de proteínas e para o funcionamento dos organismos vivos, assim, o seu estudo tem aplicações diretas na medicina, no desenvolvimento de tratamentos e na compreensão de doenças genéticas, além de ser crucial para a pesquisa genética.

As sequências de DNA são formadas por quatro bases nucleotídicas: Adenina (A), Timina (T), Citosina (C) e Guanina (G), já no RNA, a base Timina (T) é substituída por Uracila (U). Enquanto o DNA armazena as informações genéticas, o RNApossui a função de traduzir essa informação para a produção de proteínas.

Como volume de dados genéticos pode ser imenso, igualno caso do genoma humano, que contém cerca de 3 bilhões de pares de bases, é necessário utilizar técnicas computacionais avançadas, como a programação paralela para processar e analisar essas sequências de maneira eficiente.

Arquivos FASTA

Os arquivos FASTA são amplamente utilizados em bioinformática para representar sequências de DNA, RNA ou proteínas. O formato FASTA é simples e eficiente, contendo uma linha de descrição (que começa com o símbolo ">") seguida pelas linhas que descrevem a sequência biológica. Essa estrutura facilita o armazenamento e a leitura das sequências, sendo compatível com diversos softwares e ferramentas de análise genética.

Os arquivos FASTA podem armazenar sequências de nucleotídeos ou aminoácidos, e no contexto deste projeto, serão utilizados arquivos contendo versões modificadas do genoma humano, como o GRCh37/hg19. Esses arquivos são essenciais para a análise de sequências genéticas, pois oferecem dados completos e bem estruturados que podem ser processados para identificar mutações genéticas, estudar doenças e realizar outras pesquisas biomédicas.

Programação Paralela com MPI e OpenMP

A análise de grandes volumes de dados genéticos, como as sequências de DNA e RNA, necessita do uso de técnicas de programação paralela para reduzir o tempo de processamento e permitir a análise em tempo eficiente. Duas das tecnologias mais comuns para programação paralela em sistemas de computação de alto desempenho e que foram utilizados ao longo da disciplina de SuperComputação são o MPI (Message Passing Interface) e o OpenMP (Open Multi-Processing).

MPI (Message Passing Interface)

MPI é um padrão de comunicação utilizado para a programação paralela em sistemas distribuídos que permite que múltiplos processos em diferentes máquinas ou núcleos de processamento se comuniquem de forma eficiente, o que acaba sendo particularmente útil em sistemas de computação em cluster, onde a carga de trabalho pode ser dividida entre vários nós (máquinas) e cada nó pode processar uma parte dos dados de forma independente.

OpenMP (Open Multi-Processing)

OpenMP é uma API para programação paralela em sistemas de memória compartilhada. Ele é utilizado para dividir o trabalho entre diferentes núcleos de um único processador ou entre múltiplos processadores dentro de uma mesma máquina e ao contrário do MPI, que é usado em sistemas distribuídos, o OpenMP é mais adequado para sistemas com memória compartilhada, onde todos os processos podem acessar a mesma área de memória.

Deste modo, ambas as tecnologias são cruciais e podem ser combinadas para tornar viável a análise de grandes volumes de dados genéticos, reduzindo significativamente o tempo necessário para processar e analisar as sequências. Ao combinar MPI e OpenMP, é possível tirar proveito dos recursos de computação distribuída e paralela, maximizando a eficiência e a escalabilidade do sistema de análise.

Exercício 1: Contagem de Bases

Descrição: Implemente um programa paralelo que conte o número de ocorrências de cada base (A, T, C, G) em uma grande cadeia de DNA.

Uso do MPI:

```
int files_per_process = num_files / total_processos;
int start_index = rank * files_per_process;
int end_index = (rank + 1) * files_per_process;
```

```
MPI_Reduce(&locais_a, &globais_a, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
MPI_Reduce(&locais_t, &globais_t, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
MPI_Reduce(&locais_c, &globais_c, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
MPI_Reduce(&locais_g, &globais_g, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
```

Uso do OpenMP:

```
#pragma omp parallel for reduction(+:conta_a, conta_t, conta_c, conta_g)
for (size_t i = 0; i < sequencia.size(); ++i) {
    char base = sequencia[i];
    if (base == BASE_A) ++conta_a;
    else if (base == BASE_T) ++conta_t;
    else if (base == BASE_C) ++conta_c;
    else if (base == BASE_G) ++conta_g;
}
```

- MPI é utilizado para distribuir o trabalho entre diferentes processos (em máquinas ou núcleos diferentes) e para agregar os resultados utilizando MPI_Reduce no processo com rank 0.
- OpenMP é utilizado para paralelizar o laço de contagem das bases dentro de cada processo, distribuindo o trabalho entre múltiplas threads, com o uso de reduction para garantir que as contagens sejam somadas de maneira segura.
-

Exercício 2: Transcrição de DNA em RNA

Descrição: Desenvolva um programa que converta uma sequência de DNA em RNA. Lembre das substituições das bases nitrogenadas!

Uso do MPI:

```
MPI_Bcast(&comprimento_total, 1, MPI_UNSIGNED_LONG, 0, MPI_COMM_WORLD);
MPI_Bcast(const_cast<char*>(sequencia_dna.c_str()), comprimento_total, MPI_CHAR, 0,
MPI_COMM_WORLD);
```

```
MPI_Gather(&quantidade_envio, 1, MPI_INT, quantidade_recebida.data(), 1, MPI_INT, 0,
MPI_COMM_WORLD);
MPI_Gatherv(sequencia_rna_local.data(), quantidade_envio, MPI_CHAR,
            rank == 0 ? &sequencia_rna_global[0] : nullptr,
            rank == 0 ? quantidade_recebida.data() : nullptr,
            rank == 0 ? deslocamentos.data() : nullptr,
            MPI_CHAR, 0, MPI_COMM_WORLD);
```

Uso do OpenMP:

```
#pragma omp parallel for
for (size_t i = 0; i < sequencia_dna.length(); ++i) {
    switch (toupper(sequencia_dna[i])) {
        case 'A': sequencia_rna[i] = 'U'; break;
        case 'T': sequencia_rna[i] = 'A'; break;
        case 'C': sequencia_rna[i] = 'G'; break;
        case 'G': sequencia_rna[i] = 'C'; break;
        default: sequencia_rna[i] = 'N'; break;
    }
}
```

- MPI é usado para distribuir os dados entre os processos e agregar os resultados. O trabalho é dividido entre os processos e, no final, as sequências transcritas são reunidas no processo raiz (rank 0) usando MPI_Gatherv.
- OpenMP paraleliza a transcrição de DNA em RNA dentro de cada processo, utilizando múltiplas threads para processar diferentes partes da sequência de DNA simultaneamente.

Exercício 3: Trabalhando com aminoácidos

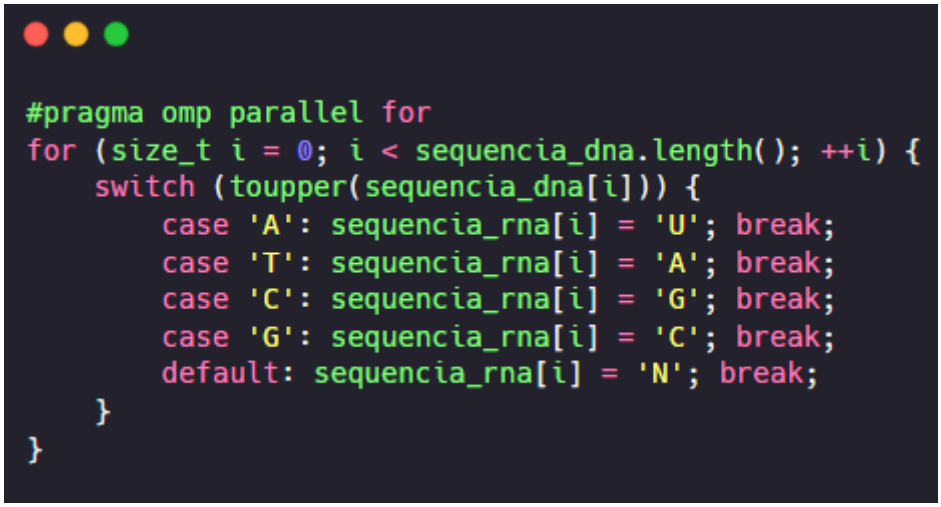
Descrição: A partir da conversão de DNA → RNA, faça um programa que conte quantas proteínas foram inicializadas (contagem de AUG). Faça a distribuição paralela que julgar pertinente.

Uso do MPI:

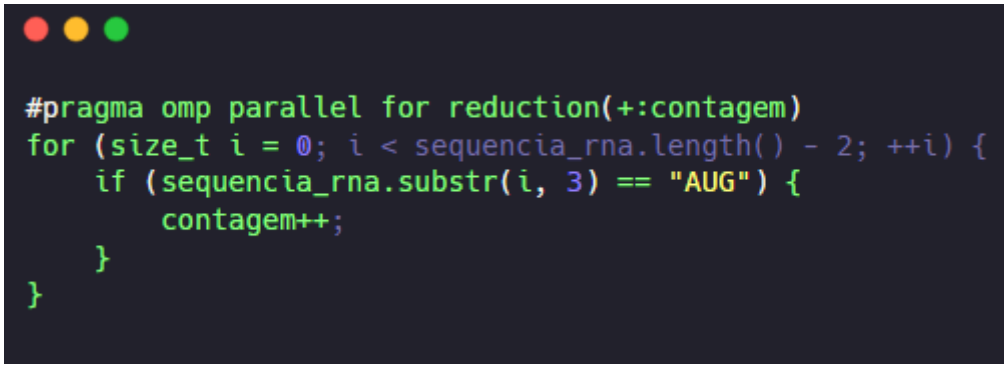
```
MPI_Bcast(&comprimento_total, 1, MPI_UNSIGNED_LONG, 0, MPI_COMM_WORLD);
MPI_Bcast(const_cast<char*>(sequencia_dna.c_str()), comprimento_total, MPI_CHAR, 0,
MPI_COMM_WORLD);
```

```
MPI_Reduce(&contagem_local, &contagem_total, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
```

Uso do OpenMP:



```
#pragma omp parallel for
for (size_t i = 0; i < sequencia_dna.length(); ++i) {
    switch (toupper(sequencia_dna[i])) {
        case 'A': sequencia_rna[i] = 'U'; break;
        case 'T': sequencia_rna[i] = 'A'; break;
        case 'C': sequencia_rna[i] = 'G'; break;
        case 'G': sequencia_rna[i] = 'C'; break;
        default: sequencia_rna[i] = 'N'; break;
    }
}
```



```
#pragma omp parallel for reduction(+:contagem)
for (size_t i = 0; i < sequencia_rna.length() - 2; ++i) {
    if (sequencia_rna.substr(i, 3) == "AUG") {
        contagem++;
    }
}
```

- MPI é utilizado para distribuir a carga de trabalho entre múltiplos processos, realizar o broadcast dos dados e agregar os resultados com MPI_Reduce. Ele permite que a sequência de DNA seja dividida entre os processos para otimizar o tempo de execução e obter uma contagem total de códons AUG.
- OpenMP paraleliza operações internas a cada processo, como a transcrição de DNA para RNA e a contagem dos códons AUG. Isso aumenta a eficiência dentro de cada processo, dividindo as tarefas entre múltiplas threads.

Exercício 4: Trabalhando com síntese proteica

Descrição: Desenvolva um programa que percorra uma sequência de RNA e traduza cada códon em seu aminoácido correspondente, até encontrar um códon de parada

Uso do MPI:

```
MPI_Bcast(&comprimento_total, 1, MPI_UNSIGNED_LONG, 0, MPI_COMM_WORLD);
MPI_Bcast(const_cast<char*>(sequencia_rna.c_str()), comprimento_total, MPI_CHAR, 0,
MPI_COMM_WORLD);
```

```
MPI_Gather(&tamanho_local, 1, MPI_INT, tamanhos_processos.data(), 1, MPI_INT, 0,
MPI_COMM_WORLD);

MPI_Gatherv(proteina_local.data(), tamanho_local, MPI_INT,
            proteina_global.data(), tamanhos_processos.data(),
            deslocamentos.data(), MPI_INT, 0, MPI_COMM_WORLD);
```

- MPI foi usado para distribuir o trabalho entre múltiplos processos e coletar os resultados de maneira eficiente. Isso permite que o programa processe grandes sequências de RNA em paralelo, utilizando múltiplos nós.